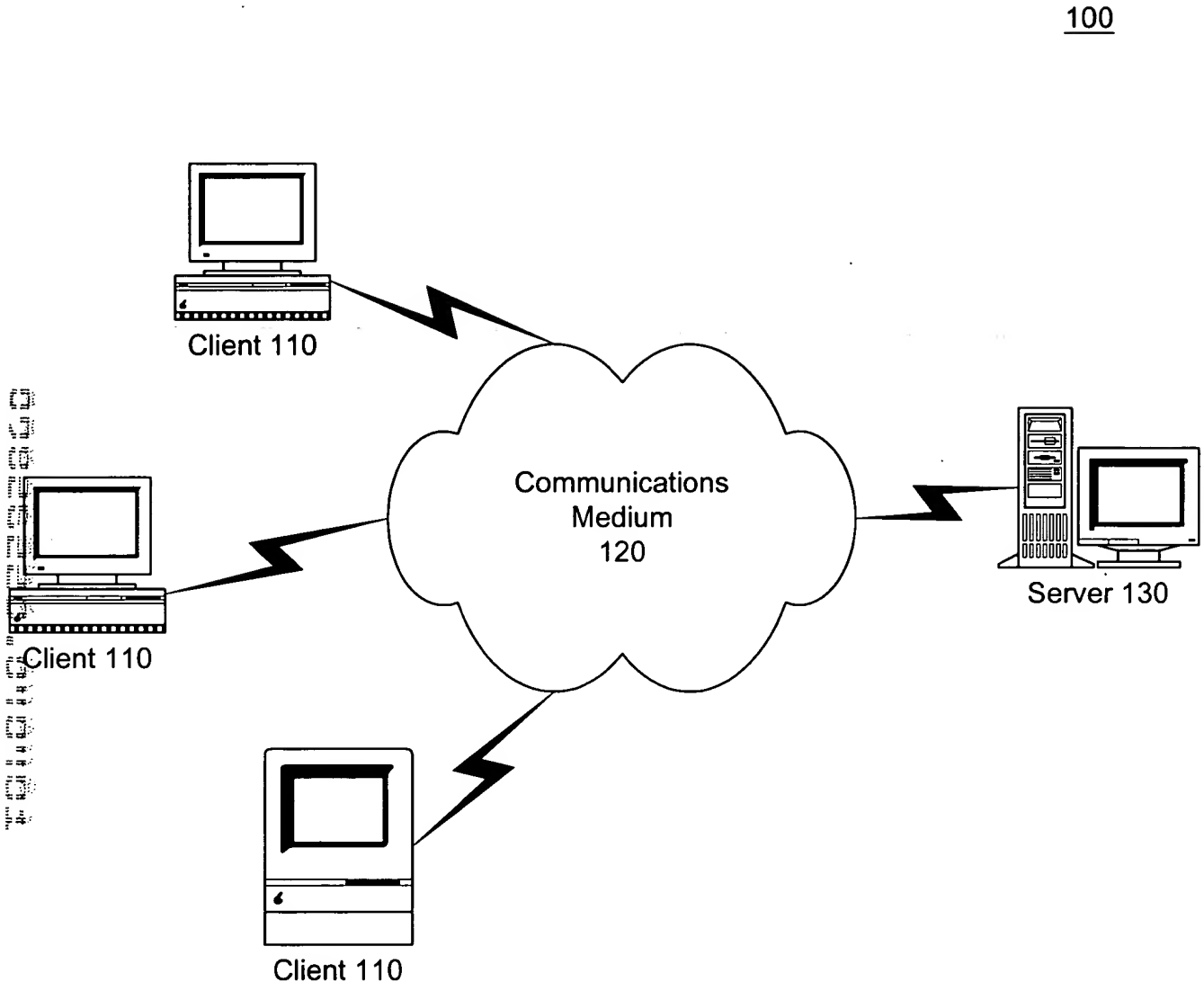
100
Client 110
Communications
Medium
120
Server 130
Client 110
Client 110

Figure 1

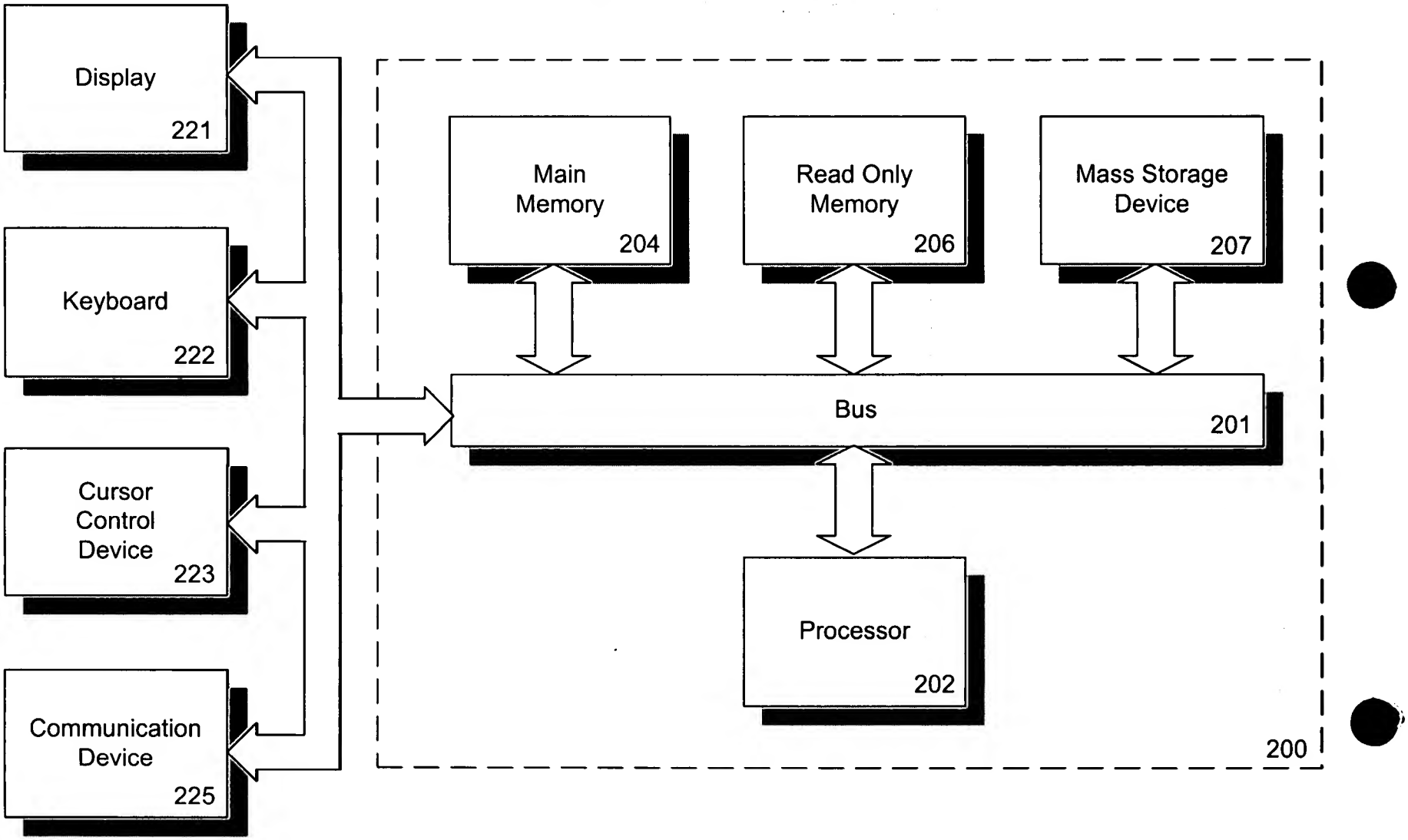
Display
221
Keyboard
222
Cursor
Control
Device
223
Communication
Device
225
Main
Memory
204
Read Only
Memory
206
Mass Storage
Device
207
Bus
201
Processor
202
200

Figure 2

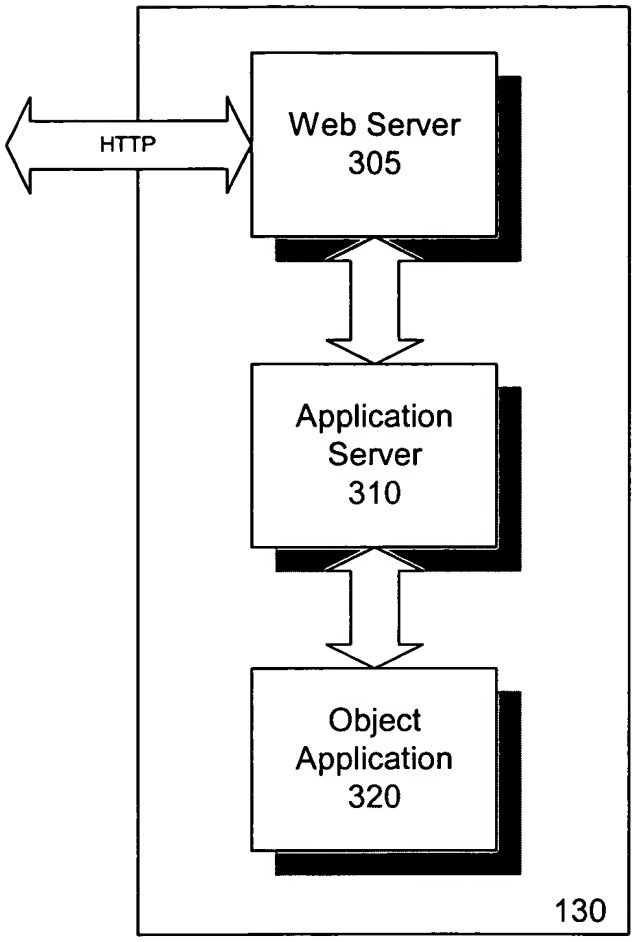
HTTP
Web Server
305
Application
Server
310
Object
Application
320
130

Figure 3

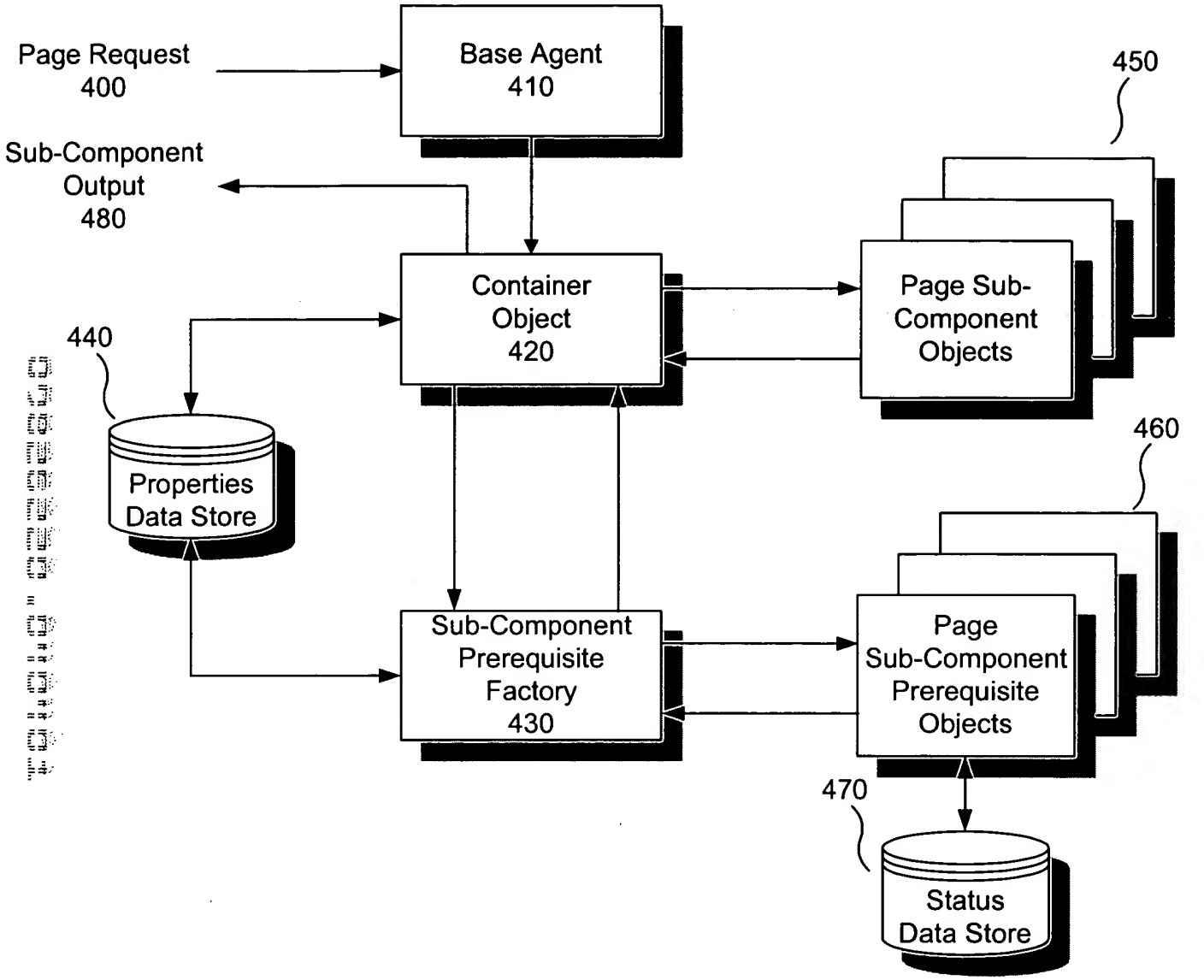
Page Request
400
Base Agent
410
450
Sub-Component
Output
480
Container
Object
420
Page Sub-
Component
Objects
440
Properties
Data Store
460
Sub-Component
Prerequisite
Factory
430
Page
Sub-Component
Prerequisite
Objects
470
Status
Data Store

Figure 4

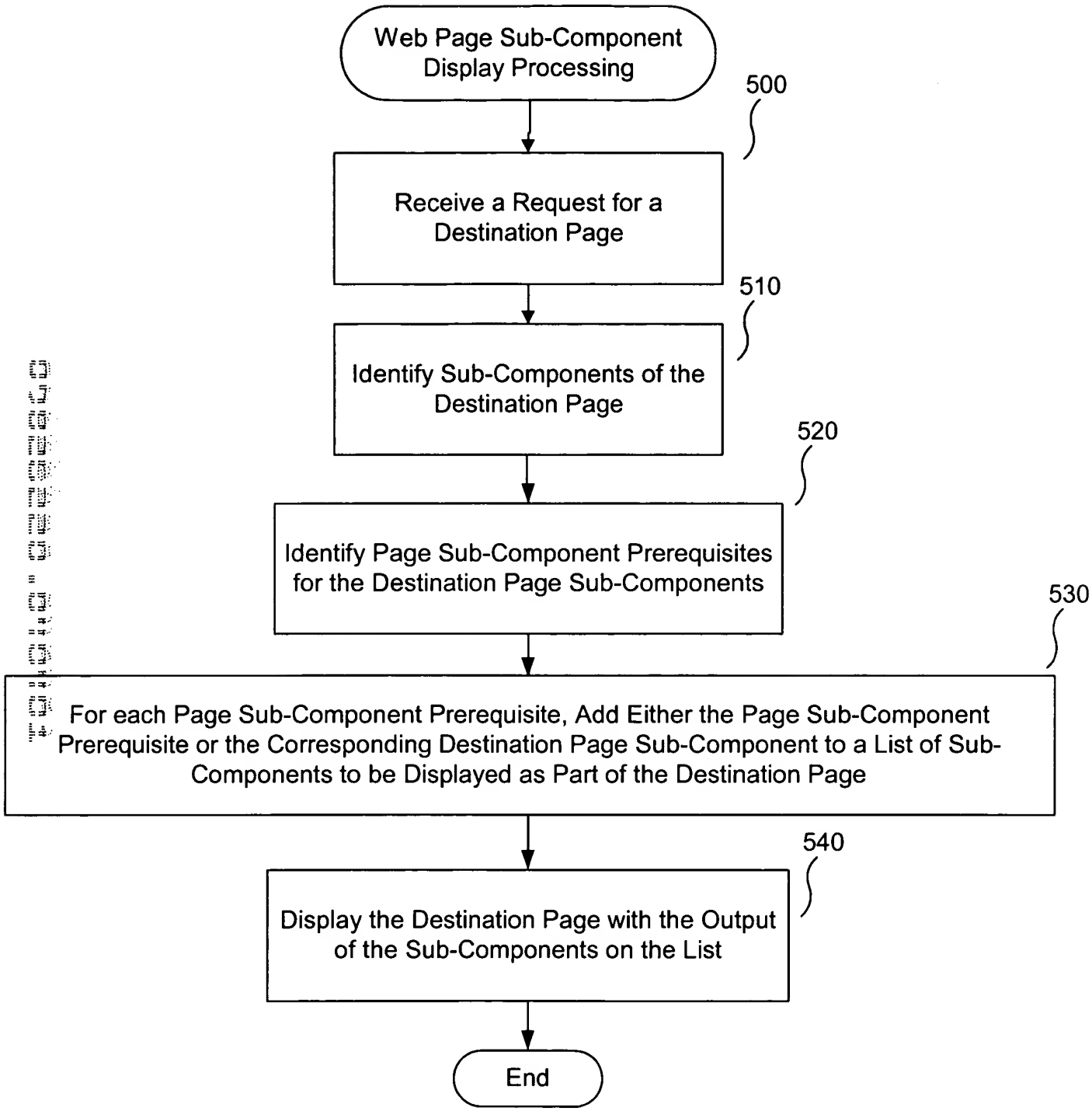
Web Page Sub-Component Display Processing
500
Receive a Request for a Destination Page
510
Identify Sub-Components of the Destination Page
520
Identify Page Sub-Component Prerequisites for the Destination Page Sub-Components
530
For each Page Sub-Component Prerequisite, Add Either the Page Sub-Component Prerequisite or the Corresponding Destination Page Sub-Component to a List of Sub-Components to be Displayed as Part of the Destination Page
540
Display the Destination Page with the Output of the Sub-Components on the List
End

Figure 5A

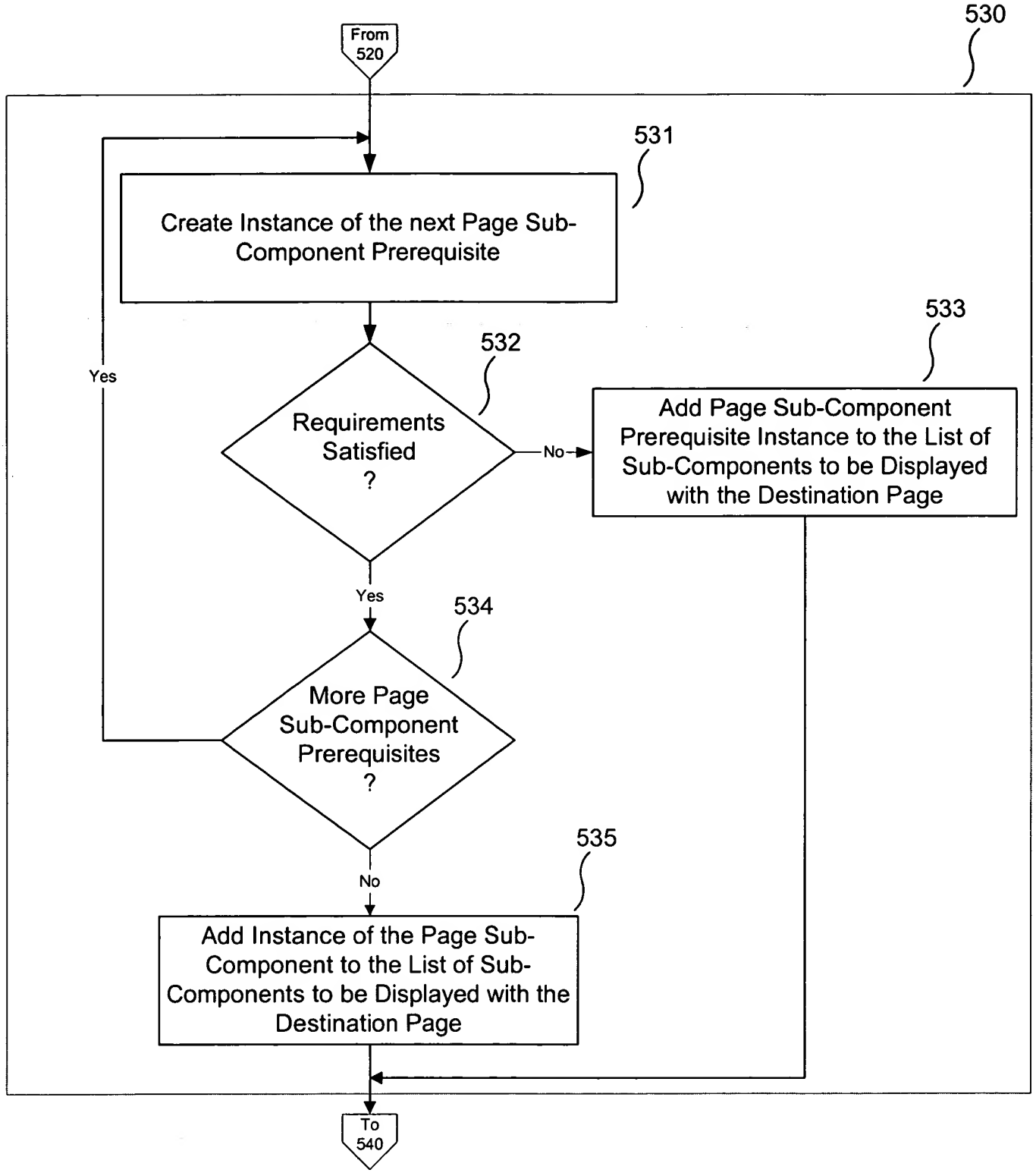
530
From 520
531
Create Instance of the next Page Sub-Component Prerequisite
532
Requirements Satisfied ?
Yes
No
533
Add Page Sub-Component Prerequisite Instance to the List of Sub-Components to be Displayed with the Destination Page
Yes
534
More Page Sub-Component Prerequisites ?
No
535
Add Instance of the Page Sub-Component to the List of Sub-Components to be Displayed with the Destination Page
To 540

Figure 5B

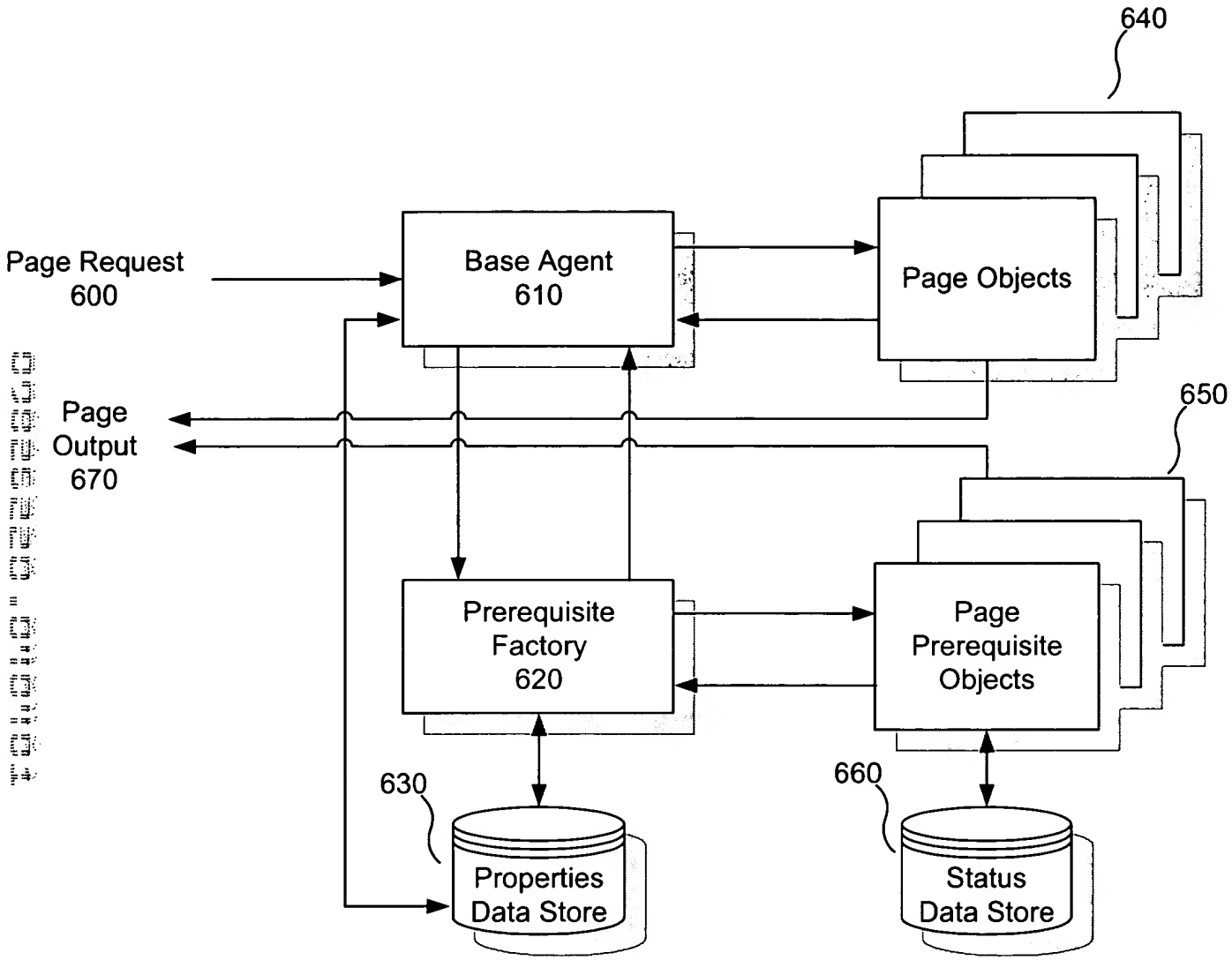
640
Page Request
600
Base Agent
610
Page Objects
Page
Output
670
650
Prerequisite
Factory
620
Page
Prerequisite
Objects
630
660
Properties
Data Store
Status
Data Store

Figure 6

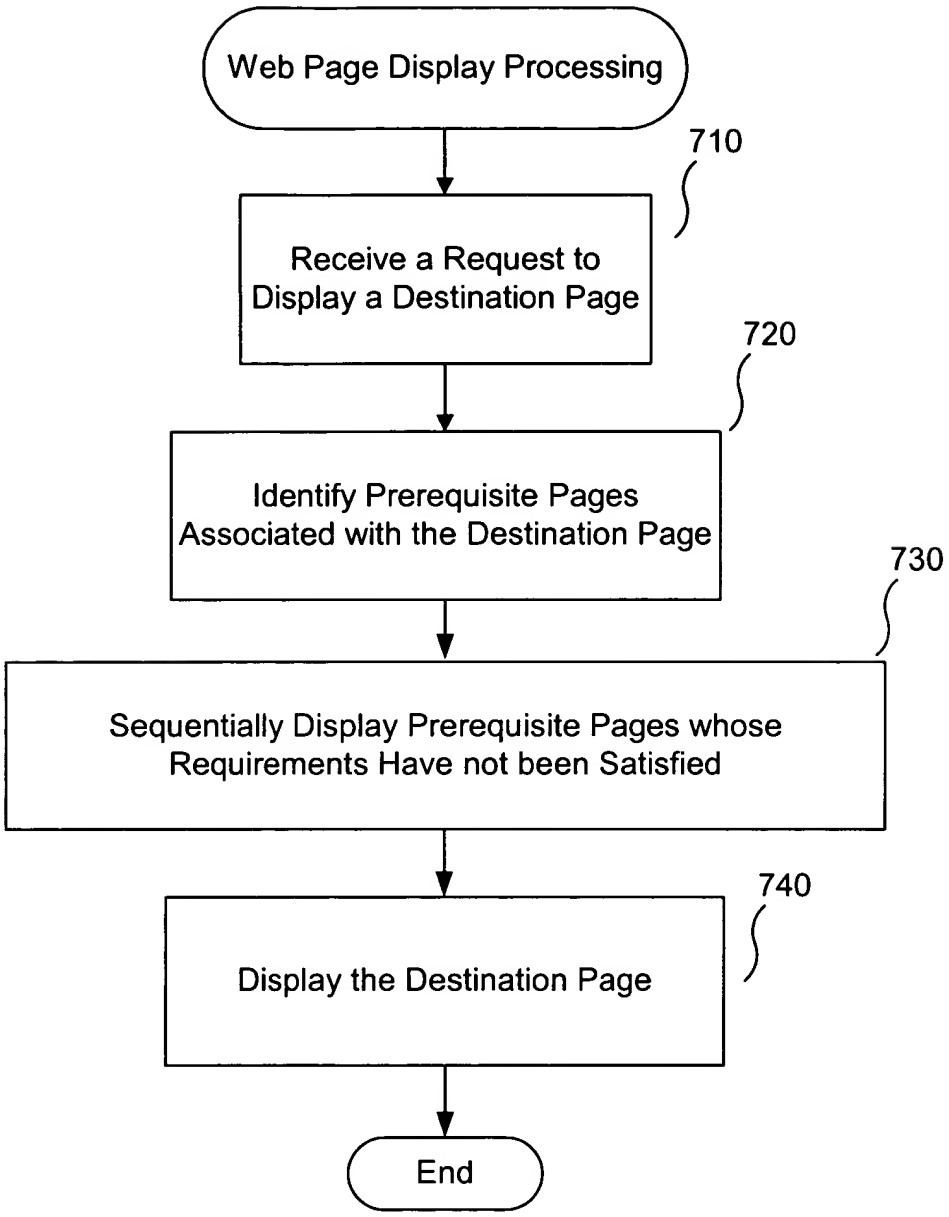
Web Page Display Processing
710
Receive a Request to Display a Destination Page
720
Identify Prerequisite Pages Associated with the Destination Page
730
Sequentially Display Prerequisite Pages whose Requirements Have not been Satisfied
740
Display the Destination Page
End

Figure 7A

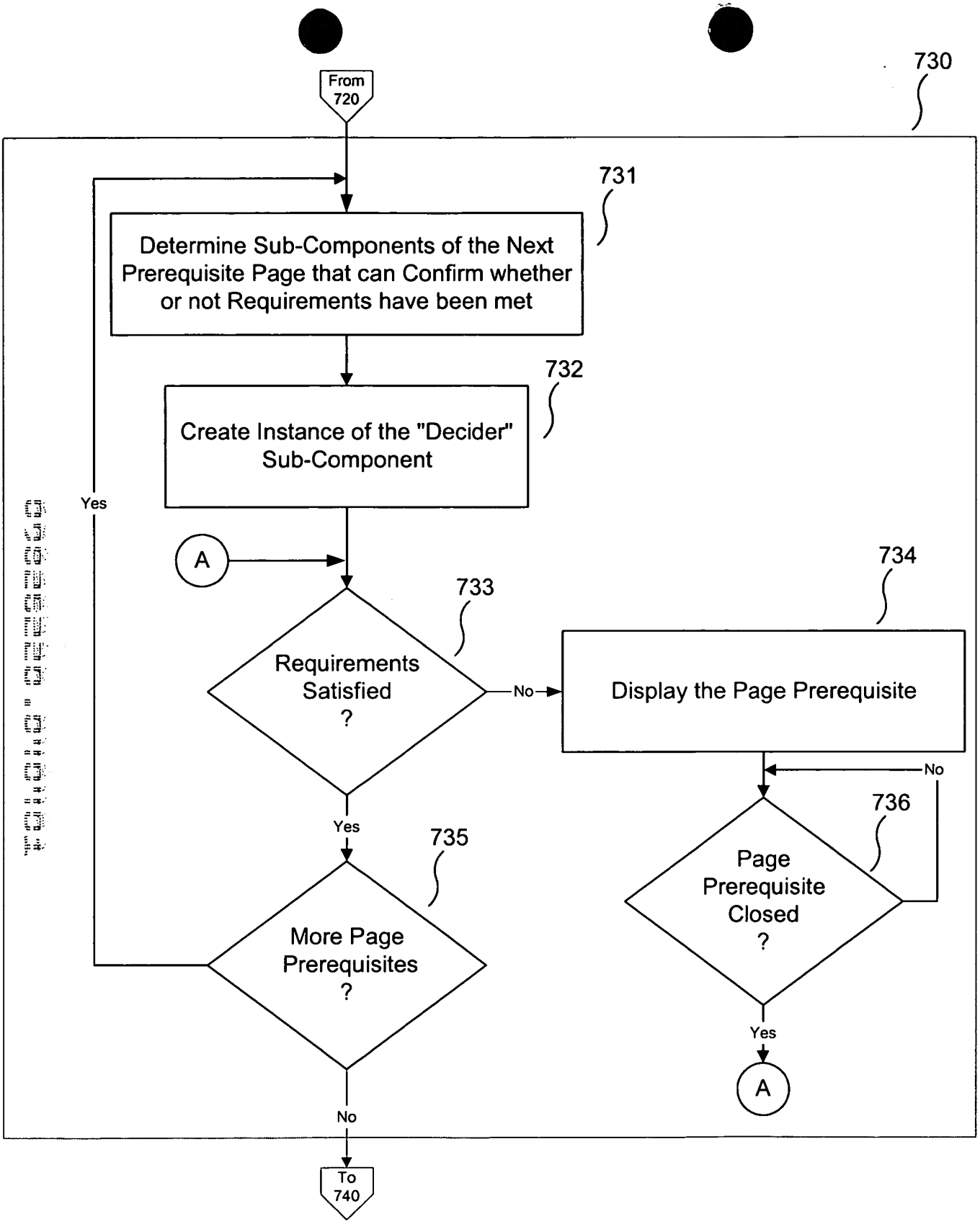
730
From 720
731
Determine Sub-Components of the Next Prerequisite Page that can Confirm whether or not Requirements have been met
732
Create Instance of the "Decider" Sub-Component
Yes
A
733
Requirements Satisfied ?
No
734
Display the Page Prerequisite
No
736
Page Prerequisite Closed ?
Yes
735
More Page Prerequisites ?
Yes
A
No
To 740

Figure 7B

<Attribute> = <Value> 810

promtp1 = please enter your name 820

Figure 8 (Prior Art)

<Object>.<Attribute> = <Value> 910

Obj1.attr1 = val11 920
Obj1.attr2 = val12 930
Obj2.attr1 = val21 940
Obj2.attr3 = val23 950

Trading.prompt1 = please enter your name 960

Figure 9

<Object>.extends = <anotherObject> 1010

Obj1.attr1 = val11 1020
Obj1.attr2 = val12 1030

Obj2.extends=Obj1 1040
Obj2.attr2 = val22 1050
Obj2.attr3 = val23 1060

Figure 10

<Context>.<Object>.<Attribute> = <Value> 1110

Obj1.attr1 = val11 1120
Obj1.attr2 = val12 1130
Page1.Obj1.attr2 = val112 1140

Figure 11

<Context>.extends = <anotherContext> 1210

Obj1.attr1 = val11 1220
Obj1.attr2 = val12 1230
Page1.Obj1.attr2 = val112 1240

Page2.extends = Page1 1250
Page2.Obj1.attr2 = val212 1260

Figure 12

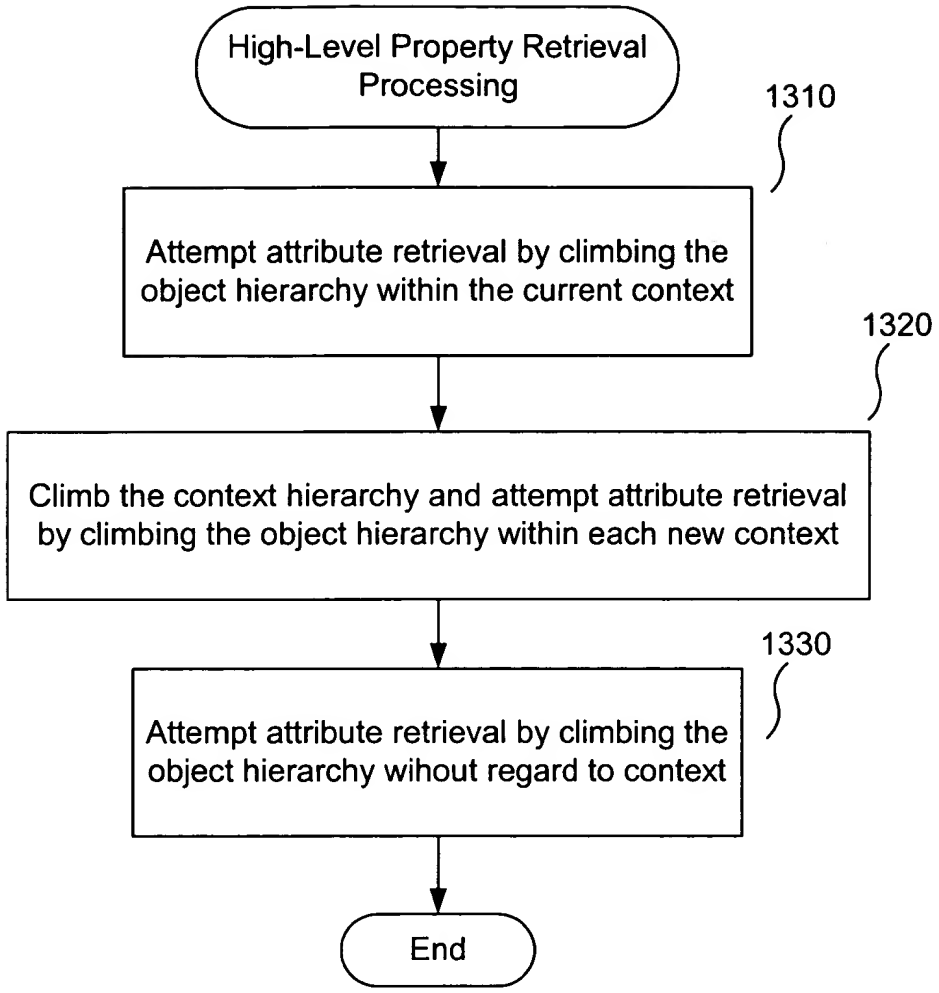
High-Level Property Retrieval Processing
1310
Attempt attribute retrieval by climbing the object hierarchy within the current context
1320
Climb the context hierarchy and attempt attribute retrieval by climbing the object hierarchy within each new context
1330
Attempt attribute retrieval by climbing the object hierarchy wihout regard to context
End

Figure 13

Page2.extends=Page1
Page3.extends=Page2

Obj2.extends=Obj1
Obj3.extends=Obj2

Figure 14A

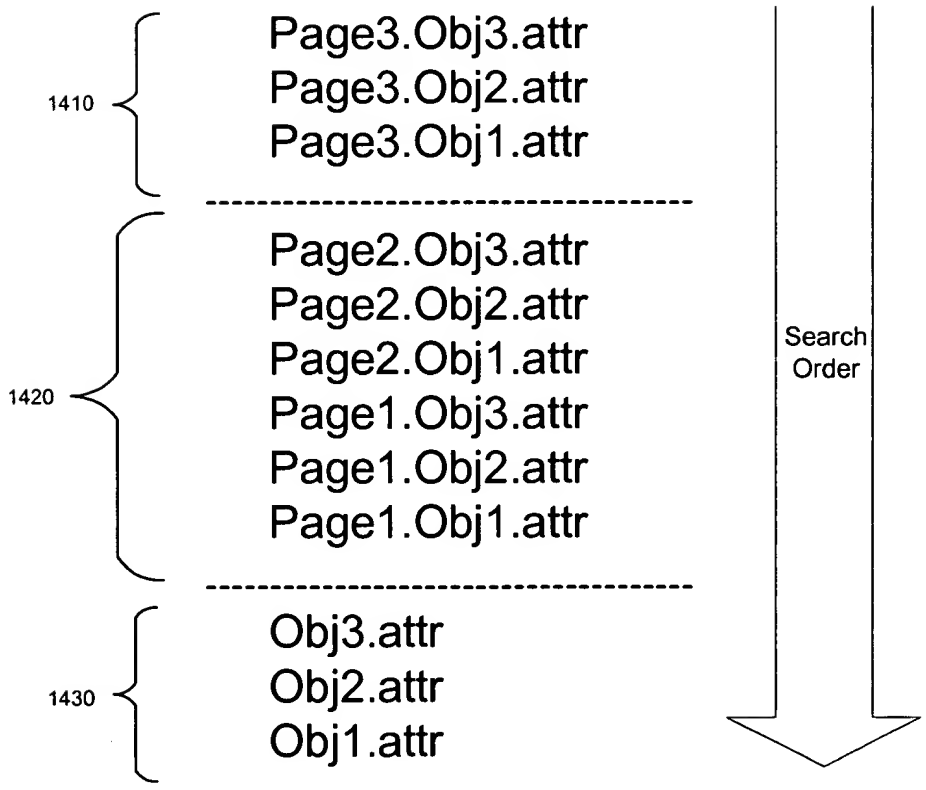

Figure 14B

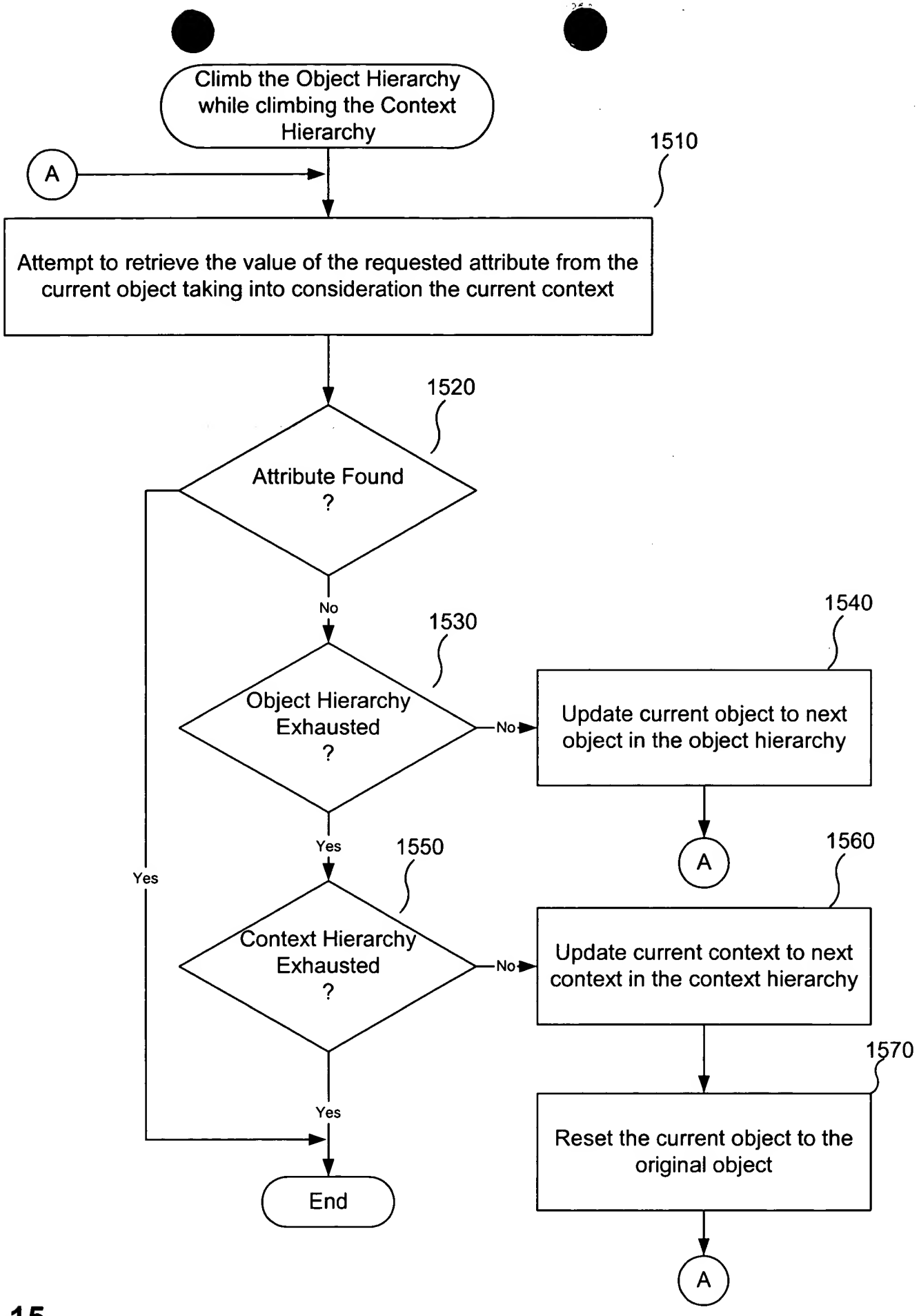
Climb the Object Hierarchy while climbing the Context Hierarchy
A
1510
Attempt to retrieve the value of the requested attribute from the current object taking into consideration the current context
1520
Attribute Found ?
No
1530
Object Hierarchy Exhausted ?
No
1540
Update current object to next object in the object hierarchy
A
Yes
1550
Context Hierarchy Exhausted ?
No
1560
Update current context to next context in the context hierarchy
1570
Reset the current object to the original object
A
Yes
Yes
End

Figure 15

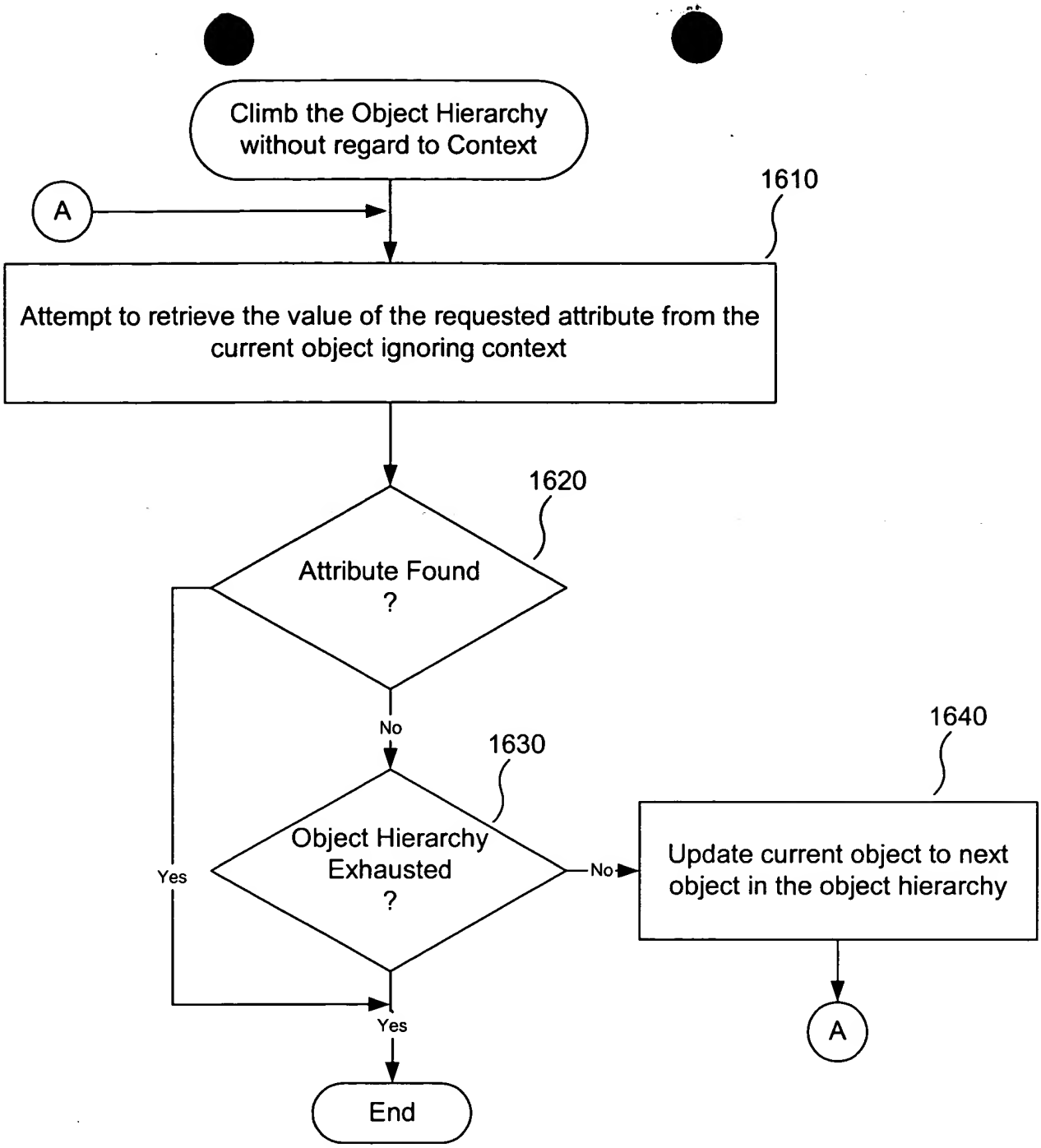
Climb the Object Hierarchy without regard to Context
A
1610
Attempt to retrieve the value of the requested attribute from the current object ignoring context
1620
Attribute Found ?
No
Yes
1630
Object Hierarchy Exhausted ?
No
Yes
1640
Update current object to next object in the object hierarchy
A
End

Figure 16

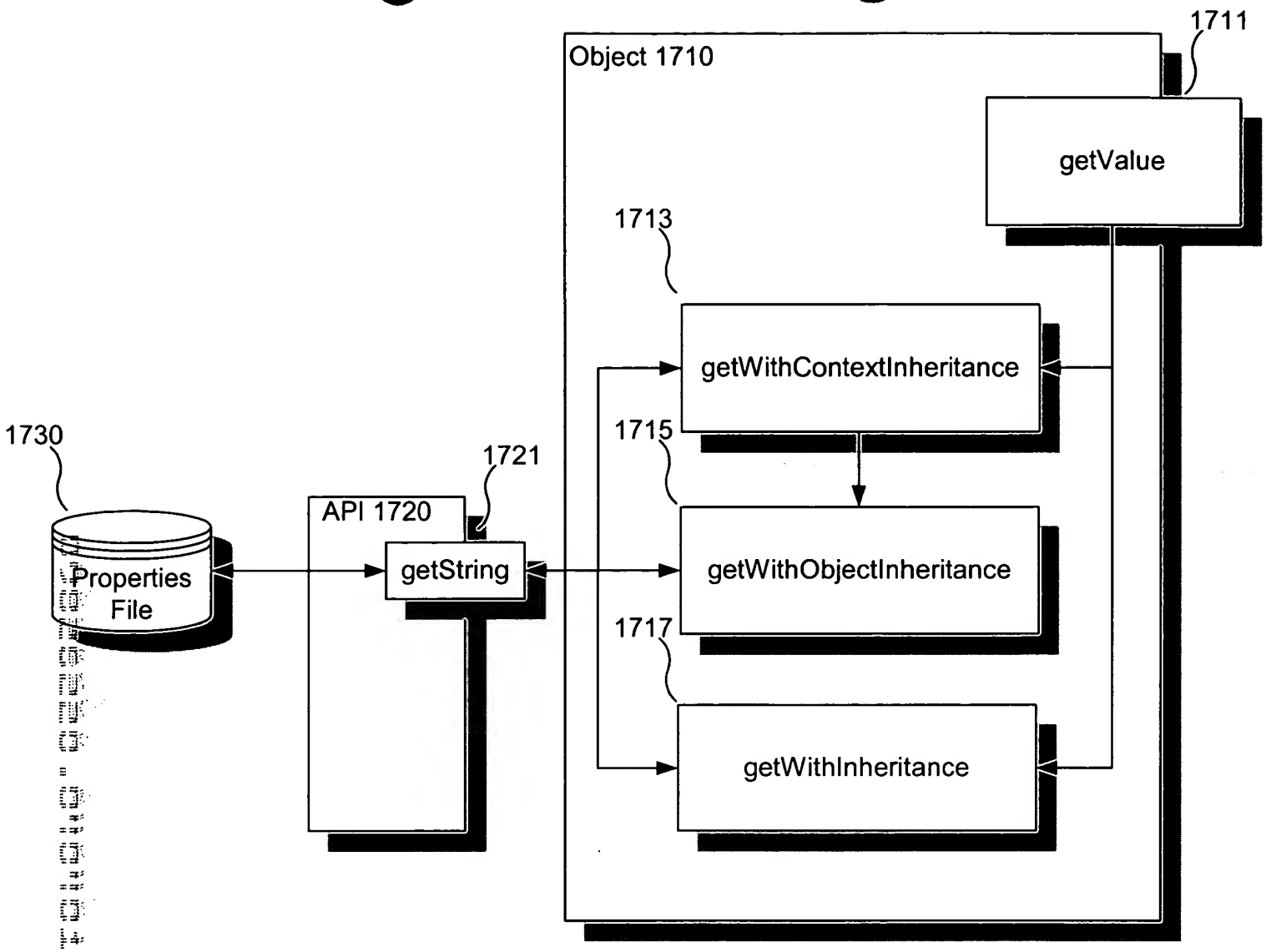
Object 1710
1711
getValue
1713
getWithContextInheritance
1715
getWithObjectInheritance
1717
getWithInheritance
1730
Properties File
API 1720
1721
getString

Figure 17

```
1.    String getValue( String attrib ) {
2.
3.          // get the container name of this object
4.          String context = getContextName();
5.
6.          // get the name of the class of this executing object
7.          String object = getClass().getName();
8.
9.          // check to see if within this context a value is defined for the
10.         // specified attribute for objects of the currently executing object's
11.         // type, or for any of its parent objects
12.
13.         String result = getWithContextInheritance( context, object, attrib );
14.
15.         if (result == null) { // if it's null it didn't find it.
16.
17.         // now try to find the value of the attribute of the object
18.         // by ignoring the context
19.
20.         // this will look for the attribute in all objects that are extended by
21.         // the current object
22.
23.               result = getWithInheritance( object, attrib );
24.
25.         } // end if
26.
27.         return result;
28.
29.   } // end getValue
```

Figure 18

```
1.    String getWithContextInheritance(
2.          String context, String object, String attrib) {
3.
4.          // check if the attrib is found in the current context by climbing up
5.          // the object hierarchy if necessary
6.          String result = getWithObjInheritance( context, object, attrib );
7.
8.          if (result == null) { //didn't find it in the current context
9.                // check if this context extends another context
10.               context = resourceBundle.getString( context + ".extends" );
11.
12.               //context now contains the name of the parent context, if any
13.               if (context != null) // found a parent context
14.                     // this is a recursive call that looks for the attribute
15.                     // and climbs up the context hierarchy
16.                     result = getWithContextInheritance( context,  object, attrib );
17
18.               } //end if
19.
20.         return result;
21.
22.   } // end getWithContextInheritance
```

Figure 19

```
1.    String getWithObjInheritance( String context, String object, String attrib ) {
2.          // concatenate context, ".", the object, ".", and the attrib
3.          String lookFor = context + "." + object + "." + attrib;
4.
5.          // check if the combined attribute is found in the resource bundle
6.          String result = resourceBundle.getString( lookFor );
7.
8.          if (result == null) { // didn't find it in the current object
9.                // check if object extends another object, meaning
10.               // it has a parent.
11.               object = resourceBundle.getString( object + ".extends" );
12.
13.               //object now contains the name of the parent, if any
14.               if (object != null) // found a parent object
15.                     // this is a recursive call that climbs up the
16.                     // object's hierarchy while keeping the context unchanged.
17.                     result = getWithObjInheritance( context, object, attrib );
18.
19.         } // end if
20.
21.         return result;
22.
23.   } // end getWithObjInheritance
```

Figure 20

```
1.    String getWithInheritance( String object, String attrib ) {
2.          // concatenate the object, ".", and the attrib
3.          String lookFor = object + "." + attrib;
4.
5.          // check if the combined attribute is found in the resource bundle
6.          String result = resourceBundle.getString( lookFor );
7.          if (result == null) { //didn't find it in the current object
8.                // check if object extends another object, meaning
9.                // it has a parent.
10.               object = resourceBundle.getString( object + ".extends" );
11.
12.               // object now contains the name of the parent, if any
13.               if (object != null) // found a parent object
14.                     // this is a recursive call. Look for the attribute
15.                     // in the parent or its parents
16.                     result = getWithInheritance( object, attrib );
17.
18.         } // end if
19.
20.         return result;
21.
22.   } // end getWithInheritance
```

Figure 21